PATENT
450100-05022

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

TITLE: ENCODER AND ENCODING METHOD

INVENTOR: Haruo TOGASHI

William S. Frommer
Registration No. 25,506
FROMMER LAWRENCE & HAUG LLP
745 Fifth Avenue
New York, New York 10151
Tel. (212) 588-0800

ENCODER AND ENCODING METHOD

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

The present invention relates to an encoder and an encoding method which are suitably applied to encoders and encoding methods according to, for example, the standard of Joint Photographic Experts Group (JPEG) 2000.

DESCRIPTION OF THE RELATED ART

A compression method called JPEG2000 has been standardized as a new data compression method.

In the encoding method according to the standard of JPEG2000, the encoding processing is complicated. Hence, there have been demands for shorter transfer periods.

SUMMARY OF THE INVENTION

The present invention has been made in view of the above situation, and proposes an encoder and an encoding method which are capable of remarkably improving the transfer efficiency necessary for encoding processings.

To achieve the above object, an encoder according to an aspect of the present invention comprises: a filtering generation means which generates a filtering coefficient by performing a filtering processing on inputted picture data; a division means

which divides the filtering coefficient into plural bit planes from an uppermost bit to a lowermost bit of each pixel; a read control means which removes a predetermined number of bit planes among the plural bit planes, from a lower side, thereafter reads remaining bit planes, and outputs the remaining bit planes in parallel; and plural encoding means which respectively perform encoding processings on the plural bit planes outputted in parallel from the read control means, wherein the read control means determines the number of the removed bit planes, so that a quantity of generated codes per frame is kept constant when each of the plural encoding means performs the encoding processing.

As a result of this, a predetermined number of bit planes corresponding to those parts of information that seem relatively small to users are removed, and accordingly, the encoder can prevent variation of the processing time in a stage before the encoding means respectively perform encoding processings.

Also, an encoding method according to another aspect of the present invention comprises: a first step of generating a filtering coefficient by performing a filtering processing on inputted picture data; a second step of dividing the filtering coefficient into plural bit planes from an uppermost bit to a lowermost bit of each pixel; a third step of removing a predetermined number of bit planes among the plural bit planes, from a lower side, thereafter reading remaining bit planes, and outputting the remaining bit planes in parallel; and a fourth step of performing encoding

processings respectively on the plural bit planes outputted in parallel, wherein the number of the removed bit planes is determined in the third step so that a quantity of generated codes per frame is kept constant when the encoding processings are performed in the fourth step.

As a result of this, a predetermined number of bit planes corresponding to those parts of information that seem relatively small to users are removed, and accordingly, it is possible for the encoding method to prevent variation of the processing time in a stage before respective encoding processings are performed.

The nature, principle and utility of the invention will become more apparent from the following detailed description when read in conjunction with the accompanying drawings in which like parts are designated by like reference numerals or characters.

BRIEF DESCRIPTION OF THE DRAWINGS

In the accompanying drawings:

FIG. 1 is a block diagram showing the structure of an encoder according to an embodiment;

FIG. 2 is a schematic plan views for explaining wavelet conversion;

FIGS. 3A and 3B are schematic diagrams for explaining code blocks and bit planes;

FIGS. 4A to 4C are schematic diagrams for explaining code blocks after quantization and the contents of bit planes

corresponding to the code blocks;

FIG. 5 is a schematic diagram for explaining functional blocks of a CPU;

FIG. 6 is a flowchart for explaining a significant sample calculation processing procedure;

FIG. 7 is a flowchart for explaining code quantity estimation and a primary rate control processing procedure;

FIG. 8 is a plan view for explaining quantities of generated codes in respective bit planes accumulated for one frame;

FIG. 9 is a plan view for explaining quantities of generated codes in respective bit planes accumulated for one frame;

FIG. 10 is a plan view for explaining quantities of generated codes in respective bit planes accumulated for one frame;

FIG. 11 is a plan view for explaining quantities of generated codes in respective bit planes accumulated for one frame;

FIG. 12 is a plan view for explaining quantities of generated codes in respective bit planes accumulated for one frame; and

FIG. 13 is a flowchart for explaining a generated code quantity estimation processing procedure.

DETAILED DESCRIPTION OF THE INVENTION

Preferred embodiment of this invention will be described with reference to the accompanying drawings:

(1) Structure of Encoder According to Present Embodiment

In FIG. 1, the reference symbol 1 denotes an encoder. When

motion-picture data D1 (e.g., standard motion-picture signals according to ITU-R BT656) is externally inputted, a video input section 2 (VIN: Video In) separates components data (luminance Y and color differences Cb and Cr) time-superposed on the motion-picture data, under control from a CPU 3. The section 2 then extracts picture-encoding areas respectively from the components data D2, and sends the areas to a register R1E.

The register R1E once stores the data component D2 supplied from the video input section 2, and also makes a synchronous dynamic random access memory (SDRAM) 4 store each data component D2 decomposed for every one frame of the original picture data via an interface PSG. At this time, the register R1E is designed to generate addresses in the SDRAM 4, in every one of frames for every data component D2.

Subsequently, the CPU 3 reads every data component D2 stored in units of frames in the SDRAM 4, and sends the data components to a wavelet converter (DWT: Discrete Wavelet Transform) 5 via the interface PSG and a register R2E.

At this time, the register R2E manages order-of-picture information necessary for wavelet conversion which will be described later, and generates addresses in SDRAM 4 in every one of frames, for every data component D2.

The wavelet converter 5 has a dual pass filter (not shown) which serves both as a low-frequency pass type and a high-frequency pass type. Performed by use of the dual pass filter is a processing

of dividing pictures based on each data component D2, by both of the frequency and time in each of the horizontal and vertical direction (i.e., wavelet conversion processing).

That is, the wavelet conversion processing is a processing as follows. An operation of extracting a particular frequency band, simultaneously suppressing the low frequency at a ratio of 2:1, and executing filtering again is repeated so that the low frequency is divided recursively. Thus, frequency components are extracted step by step from the higher-frequency side (each step will hereinafter called a level).

As a result of this, as shown in FIG. 2, the picture based on each data component D2 is divided, for every one of levels (up to the level 5 in the present embodiment), into: components (hereinafter called LH components) in which the horizontal direction corresponds to the low-frequency side and the vertical direction corresponds to the high-frequency side; components (hereinafter called HL components) in which the horizontal direction corresponds to the high-frequency side and the vertical direction corresponds to the low-frequency side; components (hereinafter called HH components) in which the horizontal direction corresponds to the high-frequency side and the vertical direction corresponds also to the high-frequency side; and components (hereinafter called LL components) in which the horizontal direction corresponds to the low-frequency side and the vertical direction corresponds to the low-frequency side.

Data components D3 in which pictures have been divided into LH, HL, HH, and LL components for every level are quantized by a quantizer (Q: quantum) 6, and then supplied to a bit-plane converter (WTB: Word to Bitplane) 7. The LL components of each data component D3 except for the final step level are stored into the SDRAM 4 via a register R4E and the interface PSG directly, i.e., without being subjected to the processings made by the quantizer 6 and the bit plane converter 7.

As shown in FIG. 3A, the bit-plane converter 7 divides each data component D3 subjected to the wavelet conversion, into units of code blocks CB each which consists of, for example, 64 × 64 (=4096) words, for every step level and for every frequency band.

Further, the bit-plane converter 7 divides each code block CB (FIG. 3A) into 16 bit planes BP0 to BP15 where the plane BP15 is constituted by a collection of uppermost bits (bit15) of respective pixels and the plane BP0 is constituted by a collection of lowermost bits (bit0) of respective pixels. At this time, the bit-plane converter 7 detects a bit plane of 0 bit (hereinafter called a zero-bit plane) for each code block with respect to each data component.

Thereafter, the CPU 3 makes the register R4E store once every code block outputted from the bit-plane converter 7 while making the SDRAM 4 sequentially store the code blocks via the interface PSG.

Subsequently, the CPU 3 reads each bit plane stored in the

SDRAM 4 while sending the bit planes in parallel, respectively to corresponding bit model sections (CBM: Coefficient Bit Modeling) 8A (8A0 to 8Ax (X=15 in the present embodiment)) and subsequently to arithmetic encoders (AC: Arithmetic) 9A (9A0 to 9Ax).

At this time, the number of those bit planes (hereinafter called processing bit planes) that have actually the bit 1 among the bit planes read from the SDRAM 4 differs for every code block. Hence, the CPU 3 is designed to control input/output states of a register R5E, in order to prevent the processing time of each of the bit model sections 8A corresponding respectively to the code blocks from varying.

The code blocks CB after quantization are each constituted by 16 bit planes, as shown in FIG. 4A. Also as shown in this figure, each code block CB includes a bit plane BP 15 (hereinafter called a code bit plane) which indicates whether the uppermost bit is positive or negative, lower several zero bit planes BPZ, and much lower several processing bit planes BPY.

Further, the CPU 3 monitors the bit planes sent from the bit-plane converter 7 via the register R4E while executing a rate control so that only necessary bit planes are sent to the register R5E.

The bit model sections 8A create contexts corresponding respectively to the bits of each bit plane in each code block supplied in parallel from the register R5E, and thereafter sends the contexts respectively to the corresponding arithmetic encoders

9A in synchronization with the bits.

The bit model sections 8A output respectively the bits and corresponding contexts, in the order in which arithmetic encoding is carried out by the arithmetic encoders 9A.

The arithmetic encoders 9A are designed to perform entropy encoding according to the JPEG2000 standard, independently for every bit plane. Therefore, the CPU 3 settles the numbers of necessary bit planes to be supplied respectively to the bit model sections 8A, in order to encode only the least necessary bit planes.

That is, among the above-mentioned 16 bit planes shown in FIG. 4B, several lowest processing bit planes BPY are regarded as non-processing bit planes BPW (FIG. 4C), so that only the remaining processing bit planes BPX (X: Y-W) are regarded as targets to be processed.

Further, the arithmetic encoders 9A calculate generation probabilities for the contexts, respectively, with respect to the bits sent from the corresponding bit model sections 8A to create encoded streams.

The arithmetic encoding method of the present embodiment differs from generally used arithmetic encoding methods in the following points. First, inputted bits are of "0" or "1". Second, the inputted bits are not encoded but More Probable Symbol (MPS) and Less Probable Symbol (LPS) are encoded. Third, the generation probabilities of the inputted bits are learned in accordance with the inputted bits. Fourth, multiplications for divisions into

segments are approximated by additions.

Thereafter, the CPU 3 stores once the code blocks after the encoding, which are respectively outputted from the arithmetic encoders 9A, into a register R6E while making the SDRAM 4 store the code blocks as encoded streams, via the interface PSG.

At this time, the time required for the encoding processing differs among the arithmetic encoders 9A. Therefore, the CPU 3 monitors the arithmetic encoders 9A and controls the register R6E to store one after another of every particular encoded stream created.

Subsequently, the CPU 3 reads only those encoded streams that are determined by a rate control from the encoded streams stored in the SDRAM 4 while sending those streams to a format generator (FMT: format) 10 via the interface PSG and a register R7E.

The encoded streams read from the SDRAM 4 by the CPU 3 via the register R6E are code strings subjected to entropy coding. Therefore, the format generator 10 rearranges the encoded streams in a desired order and adds additional information (headers) to the streams, to create data streams according to the JPEG2000 standard, which are outputted to the outside.

The CPU 3 is connected to a random access memory (RAM) 12 as a work memory so that various data can be read or written upon necessities.

(2) Method of Settling The Number of Encoded Bit Planes

The bit planes which are cut off by the transfer rate control

need not be subjected to the processings in the bit model sections 8A or the arithmetic encoders 9A, as a result of the cutting off. However, the transfer rate control is performed on the basis of the quantities of generated codes which are calculated by the arithmetic encoders 9A. Also, the quantities of generated codes can be estimated from the coefficients of respective bit planes by a simple calculation method without performing the processings in the bit model sections 8A or the arithmetic encoders 9A. The quantity of generated codes thus calculated in this method is called an estimated generated code quantity.

With respect to the estimated generated code quantities, necessary bit planes are settled to be processing bit planes by the same method as that of the transfer rate control. Naturally, the estimated generated code quantities differ from the quantities of generated codes which are obtained by performing the processings in the bit model sections 8A and the arithmetic encoders 9A. Therefore, the settled necessary bit planes with extra margins are taken as the processing bit planes. For example, provided that the hatched parts in FIG. 12 are the necessary bit planes obtained from the estimated generated code quantities, processing bit planes are determined by adding another 1 bit plane to each code block.

As described above, the CPU 3 controls the register R5E to control the number of bit planes of each code block supplied to the bit model sections 8A. Thus, the CPU 3 minimizes the processing time in each of the arithmetic encoders 9A corresponding

respectively to the bit model sections 8A.

At this time, the CPU 3 settles the number of necessary bit planes supplied to each bit model section 8A while the CPU 3 cuts off unnecessary bit planes (in the LSB side). The method of settling the number of necessary bit planes will be described below.

Specifically, the CPU 3 has an algorithm as represented by functional blocks in FIG. 5, and is constituted by a significant sample calculator 3A, a code quantity estimator 3B, a primary rate controller 3C, and a secondary rate controller 3D.

At first, the significant sample calculator 3A in the CPU 3 calculates and outputs the number of samples which become "significant" for the first time, with respect to each bit plane in each code block.

More specifically, the CPU 3 starts a significant sample calculation procedure RTI as shown in FIG. 6 from step SP0. In subsequent step SP1, the code block number cb is set to "0" and the procedure then goes to step SP2 to initialize significant [S] to "0" with respect to all samples S.

The CPU 3 then goes to step SP3 and sets the bit plane number bp to 0 as well as CountNewSig [cb][bp] to 0, where the CountNewSig [cb][bp] is the number of samples which become "significant" (a significant state, i.e., the bit which has already been "1" is taken as having been coded) for the first time at the bit plane number bp at the code block number cb. Thereafter, the CPU 3 goes to step SP4. The bit plane numbers bp are assigned in the order

from the MSB side toward the LSB side.

In step SP4, the CPU 3 selects one unprocessed sample S at the code block number cb and at the bit plane number bp, and thereafter goes to step SP5, to determine whether the bit whose significant [S] is 0 and whose bit plane number in the sample S is bp is "1" or not.

If a positive result is obtained in step SP5, the CPU 3 goes to step SP6 and sets the significant [S] to 1 and adds 1 to the CountNewSig [cb][bp]. Thereafter, the CPU 3 goes to step SP7. Otherwise, if a negative result is obtained in step SP5, the CPU 3 goes directly to step SP7.

Next in step SP7, the CPU 3 determines whether there is any unprocessed sample or not. If a positive result is obtained, the CPU 3 returns to step SP4 again and repeats the same processing as described above. Otherwise, if the result is negative, the CPU 3 goes to step SP8 and outputs the CountNewSig [cb][bp] to the code quantity estimator 3B in a later stage.

The CPU 3 then goes to step SP9 and determines whether there is any unprocessed bit plane. If a positive result is obtained, the CPU 3 goes to step SP10 and adds "1" to the bit plane number. The CPU 3 thereafter returns to step SP3 again and repeats the same processing as described above. Otherwise, if a negative result is obtained, the CPU 3 goes directly to step SP11, determining that all bit planes have been processed.

In step SP11, the CPU 3 determines whether there is any

unprocessed code block. If a positive result is obtained, the CPU 3 goes to step SP12 and adds "1" to the code block number. The CPU 3 thereafter returns to step SP2 and repeats the same processing as described above. Otherwise, if a negative result is obtained, the CPU goes to step SP13, determining that all code blocks have been processed, and terminates the significant sample calculation processing procedure RT1.

Thus, the significant sample calculator 3A in the CPU 3 can calculate and output the number of samples which become significant for the first time, with respect to each bit plane in each code block.

Subsequently, the code quantity estimator 3B in the CPU 3 outputs an estimated value of the quantity of codes generated in each bit plane in each code block. Further, the primary rate controller 3C in the CPU 3 is inputted with the estimated code quantity for every bit plane and performs control based on fixed bit truncation (rounding down), to output the number of necessary bit planes for all the code blocks.

Note that the secondary rate controller 3D in the CPU 3 is designed to perform rate control based on the fixed bit truncation in a manner as follows. That is, the controller 3D is inputted with not only the estimated code quantity for every bit plane but also a quantity (hereinafter called a target code quantity) to control the transfer rate in order to keep the quantity of generated codes per frame constant, as will be described later.

Specifically, the CPU 3 starts code quantity estimation and a primary rate control processing procedure RT2, as shown in FIG. 7, from step SP20. In subsequent step SP21, the CPU 3 sets the bit plane number to "0" as well as the number of available bytes (or byte amount) to a target byte amount. Thereafter, the CPU 3 goes to step SP22, and initializes the code block number cb to "0" as well as the estimated code quantity Bits [cb] to "0".

Further, the CPU 3 goes to step SP23 and determines whether the bit plane number bp is "0" or not. If a positive result is obtained, the CPU 3 goes to step SP24 and sets CountMR [cb][bp] to 0, and then goes to step SP25. Otherwise, if a negative result is obtained in step SP23, the CPU 3 goes to step SP26 and sets the CountMR [cb][bp] which expresses the number to be subjected to magnitude refinement (MR) using the bit plane number bp at the code block number cb, to a value decided by adding the CoutMR [cb][bp-1] and CountNewSig [cb][bp-1] of the preceding plane number. Thereafter, the CPU 3 goes to step SP25. Note that the CountMR [cb][bp] is defined as Σ (i=0 to bp-1) CountNewSig [cb][i] where i=0 to bp-1.

In step S25, the CPU 3 expresses a temporal estimated code quantity BitsTmp as CountNewSig [cb][bp] $\times$ RatioNewSig+CountMR[cb][bp] $\times$ RatioMR, where the RatioNewSig and RatioMR are parameters. Also, the CPU 3 converts the estimated code quantity into a byte amount by dividing the amount of available bytes (AvailableBytes) by 8 and rounding down decimals, and adds

the temporal estimated code quantity BitsTmp to the estimated code quantity Bits [cb].

Subsequently, in step SP27, the CPU 3 determines whether the amount of available bytes is 0 or less or not. If a negative result is obtained, the CPU 3 goes to step SP28, and determines whether there is any unprocessed code block or not. Otherwise, if a positive result is obtained, the CPU 3 directly goes to step SP29 and terminates the code quantity estimation and the primary rate control processing procedure RT2.

If a positive result is obtained in step SP28, the CPU 3 goes to step SP30 and adds "1" to the code block number cb. Thereafter, the CPU 3 returns to step SP23 and repeats the same processing as described above. Otherwise, if a negative result is obtained, the CPU 3 goes to step SP31, determining that all the code blocks have been processed.

In step SP31, the CPU 3 determines whether there is any unprocessed bit plane or not. If a positive result is obtained, the CPU 3 goes to step SP32 and adds "1" to the bit plane number bp. Thereafter, the CPU 3 returns to step SP22 again and repeats the same processing as described above. Otherwise, if a negative result is obtained, the CPU 3 goes directly to step SP29, determining that all the bit planes have been processed, and terminates the code quantity estimation and the primary rate control processing procedure RT2.

Thus, the code quantity estimator 3B and primary rate

controller 3C in the CPU 3 can output an estimated code quantity of codes to be generated in each bit plane in each code block, and can settle the number of bit planes necessary for all the code blocks, based on the estimated code quantity of each bit plane.

(3) Rate Control

The CPU 3 (the secondary rate controller 3D shown in FIG. 5) selects components in the order from the luminance Y to the color differences Cb and Cr, in the direction from the MSB side toward the LSB side in the order from the lower frequency side to the higher frequency side until a target code quantity is obtained for every of units of bit planes, referring to the target code quantity to control the transfer rate, in order that the quantity of generated codes per frame which are sent from the format generator 10 should be kept constant.

In the above-described method of settling the number of coded bit planes, the number of necessary bit planes is settled with respect to the estimated code quantity. However, the rate control is designed to perform control using fixed bit truncation, based on the quantities of generated codes obtained by performing the processings in the bit model sections 8A and arithmetic encoders 9A.

Specifically, the CPU 3 starts a generated code quantity estimation processing procedure RT3 shown in FIG. 13, from step SP40. In subsequent step SP41, the amount of available bytes (AvailableBytes) is set to a value obtained by subtracting the quantity of headers (e.g., main headers, tile-part headers, and

packet headers) from the target byte amount.

Note that the quantity of codes of headers needs to be calculated and estimated in advance so that the whole coded streams have a quantity within the target code quantity. In this case, the quantity of codes of packet headers cannot be settled, and hence, an approximate value is used as a prerequisite.

The CPU 3 then goes to step SP42, and sets the bit plane number bp to "0". The CPU 3 further goes to step SP43 and sets the code block number cb to "0".

Subsequently, the CPU 3 goes to step SP44 and detects the amount of bytes (cb, bp), i.e., the quantity of actually generated codes at the code block number cb and the bit plane number bp. Also, the CPU 3 subtracts the amount of bytes (cb, bp) from the amount of available bytes (AvailableBytes), and then, goes to step S45.

In step SP45, the CPU 3 determines whether the amount of available bytes is 0 or less. If a negative result is obtained, the CPU 3 goes to step SP46 and determines whether there is an unprocessed code block or not. Otherwise, if a positive result is obtained, the CPU 3 goes to step SP47.

If a positive result is obtained in step SP46, the CPU 3 goes to step SP48 and adds "1" to the code block number cb. Thereafter, the CPU 3 returns to step SP43 again and repeats the same processing as described above. Otherwise, if a negative result is obtained, the CPU 3 goes to step SP49, determining that all the code blocks have been processed.

In step SP49, the CPU 3 determines whether there is an unprocessed bit plane or not. If a positive result is obtained, the CPU 3 goes to step SP50 and adds "1" to the bit plane number bp. Thereafter, the CPU 3 returns to step SP42 again, and repeats the same processing as described above. Otherwise, if a negative result is obtained, the CPU 3 goes directly to step SP51, determining that all the bit planes have been processed. The CPU 3 then terminates the generated code quantity estimation processing procedure RT3.

In contrast, in step SP47 which follows the positive result of step SP45, the CPU 3 determines whether the amount of available bytes is 0 or not. If a negative result is obtained, the CPU 3 goes to step SP 52 and determines whether the code block number cb is "0" or not. Otherwise, if a positive result is obtained, the CPU 3 goes directly to step SP51 and terminates the generated code quantity estimation processing procedure RT3.

If the code block number cb is determined to be "0" in step SP52, the CPU 3 goes to step SP53 and subtracts "1" from the code block number cb. Thereafter, the CPU 3 goes to step SP51 and terminates the generated code quantity estimation processing procedure RT3.

Otherwise, if the code block number cb is not determined to be "0" in step SP52, the CPU 3 goes to SP54, and sets the code block number cb to a maximum value cbmax. Also, the CPU 3 subtracts "1" from the bit plane number bp. Thereafter, the CPU 3 goes to step SP51 and terminates the generation code quantity estimation

processing procedure RT3.

FIG. 8 shows a state where quantities of generated codes in each bit plane and calculated by the arithmetic encoders 9A are accumulated for one frame. The number of code blocks is expressed where bit planes are assigned to the ordinate axis as well as code blocks to the abscissa axis.

In the example of the division of code blocks shown in FIG. 2, two components at the level 3, six components at the level 2, and twenty components at the level 1 constitute total eighty six components. In FIG. 8, hatched parts are bit planes which include streams, i.e., significant bit planes in each of which the quantity of generated codes is not zero. Zero bit planes detected by the bit plane converter 7 do not generate codes.

Here, with respect to the significant bit planes (hatched parts in FIG. 8), the quantities of generated codes are integrated in the order from the bits of the MSB side and from the higher level (i.e., the left upper side in FIG. 8). If a target code quantity is exceeded as a result of continuing the integration, the bit planes are terminated at the maximum integration result which does not exceed the target code quantity. Specifically, in FIG. 9, the bit planes are terminated at the bit 3 in the ordinate axis and in the middle of the HL components at the level 1 in the abscissa axis. Otherwise, if the target code quantity is not exceeded after continuing the integration, all the bit planes are integrated. Thus, the integrated bit planes are settled as necessary bit planes

(hatched parts in FIG. 9).

In practice, the hierarchical levels and bands of respective wavelets are weighted. That is, the coefficients obtained from the wavelet converter 5 (FIG. 1) are quantized by the quantizer 6.

In another example shown in FIG. 10, quantization is performed such that LL components at the level 3 are divided by 1 (i.e., not quantized), HL and LH components at the level 3 are divided by 2, and HH components at the level 3 are divided by 4. Quantization is performed such that HL components and LH components at the level 2 are divided by 16, and HH components at the level 2 are divided by 32. Further, HL and LH components at the level 1 are divided by 128 and HH components at the level 1 are divided by 256.

Thus, the transfer rate is controlled with respect to quantized bit planes as shown in FIG. 11 corresponding to FIG. 8, and necessary bit planes are settled as shown in FIG. 12.

Thus, with respect to each bit plane in each code block, the CPU 3 selects components in the order from the luminance Y to the color differences Cb and Cr, in the direction from the MSB side toward the LSB side in the order from the lower frequency side to the higher frequency side, until a target code quantity is reached for each of units of bit planes. As a result, the rate control based on fixed bit truncation can be achieved finally.

(4) Operations and Advantages of the Embodiment

In the structure as described above, the encoder 1 divides respective data components D3 subjected to wavelet conversion, for

every hierarchical level and every frequency band, regarding each word having a predetermined size as a code block. Thereafter, each of the code blocks is divided into plural bit planes from the uppermost bit to the lowermost bit for every pixel.

At this time, data in units of code blocks which are read from the SDRAM 4 and supplied to the register R5E is subjected to rate control so as to remain significant bit planes having bits in the MSB side and a higher level among the plural bit planes, and to cut off lower bit planes (in the LSB side) to which small information is assigned.

As a result of this, the bit model sections 8A respectively create contexts corresponding to bits in the bit planes in each of code blocks supplied in parallel from the register R5E. Thereafter, the corresponding arithmetic encoders 9A perform orderly the entropy encoding processing according to the JPEG2000 standard, synchronizing the contexts respectively with the bits. At this time, the processing can be achieved in a minimum necessary time.

Further, the arithmetic encoders 9A calculate generation probabilities respectively for the contexts, with respect to the bits supplied from the corresponding bit model sections 8A. Thereafter, streams are sent to the format generator 10 via the SDRAM 4, in a manner in which the quantity of generated codes per frame is kept constant by selecting components in the order from the luminance Y to the color differences Cb and Cr, in the direction from the MSB side toward the LSB side in the order from

the lower frequency side to the higher frequency side, until a target code quantity is obtained for every of units of bit planes, based on the calculation results from the arithmetic encoders 9A, with respect to each bit plane in each code block. The format generator 10 rearranges the streams in a desired order and adds headers, to create encoded streams.

According to the structure as described above, the data transfer rate is controlled so that the quantity of generated codes per frame is kept constant, by removing those parts that seem relatively small information to users, when the encoder 1 performs the encoding processing according to the JPEG2000 standard. The time required for the encoding processing can thus be reduced to the minimum, and accordingly, the transfer efficiency in the encoding processing can be improved. Further, the quantity of generated codes is estimated prior to the bit model sections 8A and the arithmetic encoders 9A, so that the processings performed in the bit model sections 8A and the arithmetic encoders 9A can be minimized.

(5) Other Embodiments

As described above, the present embodiment has been described with respect to a case where an encoder having the structure shown in FIG. 1 is applied to an encoder according to the present invention. However, the present invention is not limited to this case but is applicable widely to other various structures as far as a predetermined number of bit planes corresponding to those parts

of information that seem relatively small to users are removed to improve the transfer efficiency in the encoding processing and the time for the encoding processing can be reduced to the minimum.

Also, the present embodiment has been described with respect to a case as follows. That is, the wavelet converter 5 shown in FIG. 1 is applied as a filtering generation means which divides data into code blocks (filtering coefficients) with use of wavelet coefficients subjected to wavelet conversion in units of frames. Thus, low frequency components are recursively and hierarchically extracted from the higher frequency side of picture data. Thereafter, the picture data is divided into respective code blocks for every hierarchical level and every frequency band. However, the present invention is not limited to this case but is applicable widely to other various structures as far as filtering coefficients can be generated by performing filtering processings on inputted picture data.

Also, the present embodiment has been described with respect to a case where the bit plane converter 7 shown in FIG. 1 is applied as a division means which divides code blocks (filtering coefficients) divided by the filtering generation means, further into plural bit planes from the uppermost bit to the lowermost bit of each pixel. However, the present invention is not limited to this case but is applicable widely to other various structures.

Also, the present embodiment has been described with respect to a case where the plural bit planes divided by the bit plane

converter (division means) 7 are stored in the SDRAM 4 (FIG. 1). However, the present invention is not limited to this case but is applicable to a storage means having other various structures.

Also, the present embodiment has been described with respect to a case where the CPU 3 is applied as a read control means which removes a predetermined number of bit planes among the plural bit planes stored in the SDRAM 4 from the lower side and thereafter reads out and outputs only the remaining bit planes respectively in parallel. However, the present invention is not limited to this case but is applicable widely to other various structures.

As described above, according to the present invention, an encoder comprises: a filtering generation means which generates a filtering coefficient by performing a filtering processing on inputted picture data; a division means which divides the filtering coefficient into plural bit planes from an uppermost bit to a lowermost bit of each pixel; a read control means which removes a predetermined number of bit planes among the plural bit planes, from a lower side, thereafter reads remaining bit planes, and outputs the remaining bit planes in parallel; and plural encoding means which respectively perform encoding processings on the plural bit planes outputted in parallel from the read control means, wherein the read control means determines the number of the removed bit planes, so that a quantity of generated codes per frame is kept constant when each of the plural encoding means performs the encoding processing. Therefore, a predetermined number of bit

planes corresponding to those parts of information that seem relatively small to users are removed, and accordingly, the encoder can prevent variation of the processing time in a stage before the encoding means respectively perform encoding processings. It is thus possible to realize an encoder capable of greatly improving the transfer efficiency in the encoding processings.

Also, according to the present invention, the encoding method comprises: a first step of generating a filtering coefficient by performing a filtering processing on inputted picture data; a second step of dividing the filtering coefficient into plural bit planes from an uppermost bit to a lowermost bit of each pixel; a third step of removing a predetermined number of bit planes among the plural bit planes, from a lower side, thereafter reading remaining bit planes, and outputting the remaining bit planes in parallel; and a fourth step of performing encoding processings respectively on the plural bit planes outputted in parallel, wherein the number of the removed bit planes is determined in the third step so that a quantity of generated codes per frame is kept constant when the encoding processings are performed in the fourth step. Therefore, a predetermined number of bit planes corresponding to those parts of information that seem relatively small to users are removed, and accordingly, it is possible for the encoding method to prevent variation of the processing time in a stage before respective encoding processings are performed. It is thus possible to realize an encoding method capable of greatly improving

the transfer efficiency in the encoding processings.

While there has been described in connection with the preferred embodiments of the invention, it will be obvious to those skilled in the art that various changes and modifications may be aimed, therefore, to cover in the appended claims all such changes and modifications as fall within the true spirit and scope of the invention.